September 27–30, 2005 • Grapevine (Dallas), Texas

2005 PASS Community Summit
Microsoft SQL Server Users Conference & Expo

504M

Upgrading the Database Engine

New Features in the SQL Server 2005 Database Engine

Don Vilen
Program Manager
SQL Server Engine
Microsoft Corporation

PASS
PROFESSIONAL ASSOCIATION FOR SQL SERVER
www.sqlpass.org

Transparent Benefits of Upgrading to SQL Server 2005

- Each feature we discuss belongs to one of the following two categories
  - No change to anything
    - For example, Progress Reporting in DBCC
  - Change instance or database configuration
    - For example, Asynchronous Statistics Auto-update

Where are They?

- Full-Text Search
- Locking
- DBCC
- Storage Engine
- SQL Server's Internal Operating System
- Query Optimization and Statistics
- Compilation and Plan Sharing
- Query Execution
- Replication
- Connectivity

Full-Text Search

- Full-text is multi-instanced
  - Management is easier
- Full-text is isolated from other instances / products
  - More predictability as binaries will not be changed underneath by the installation of another product
- In a Failover Cluster Full-text is clustered with its SQL Server and SQL Server Agent
- Index population: depending on the data, 100x faster
- Query performance: 30-50% faster; more in some cases
- Language in queries (except for blobs) – 3-tier
  - Default full-text language of the instance – sp_configure
  - If set, the Locale ID on the full-text index
  - If set, the Locale ID on the query itself
- Full-text catalogs are included in backup / restore and detach / attach

Locking

- Lock memory is more NUMA-aware
- Lock Partitioning
  - Splits a single lock resource into multiple lock resources, one per CPU

Details

Details

Lock Partitioning

- Splits a single lock resource into multiple lock resources, one per CPU
  - NL, SCH_S, IS, IU, and IX modes can be acquired on a single CPU while all other modes (S, U, X and SCH-M) must be acquired on all CPUs
  - Only full object locks (e.g., table, views, stored processes) can be partitioned
- Partitioning is turned on by default for systems with more than 16 CPUs

Lock Partitioning

Why It Is Needed

Details

- On higher-end systems (and on benchmarks) almost every connection to the server acquires an intent table lock (i.e., object lock) on a common table
  - This causes heavy contention on the spinlock that controls access to that table lock
- If we have >16 CPUs, lock contention is lowered by partitioning acquisition of locks
- With lock partitioning we distribute the load across multiple spinlocks and most accesses to any given spinlock will be from the same CPU (and practically always from the same node) meaning the spinlock-acquire should not spin often

Lock Partitioning

What You Will See

- sys.dm_tran_locks has new resource_lock_partition column
- Profiler: Partition ID of the table lock is represented in column 'BigintData1'
- Lock perf counters: For table locks (SH, X) and metadata lock, there will be one lock taken for each partition, so there will be a larger value in the counter
- Error Log: Whenever lock partitioning is active the error log will have:

  Lock partitioning is enabled. This is an informational message only. No user action is required.

DBCC

- DBCC CHECK* all use Database Snapshot framework avoiding locks
- LOB compaction in shrink and defrag
  - DBCC SHRINK*, DBCC INDEXDEFRAG, ALTER INDEX REORGANIZE
  - Reserved but unused free space for LOBs drops dramatically
- Progress reporting in DBCC CHECK*
  - DBCC CHECK*
  - Long running CHECK* operations are tracked in sys.dm_exec_requests
- Progress reporting in shrink
  - DBCC SHRINK*
  - Long running shrink operations are tracked in sys.dm_exec_requests

Storage Engine

- Rows can now be larger than a page Details
- *tempdb* scalability and best practi Details
- Backup and Restore Details
- Insert Operations Details
- Database Recovery Details
- Row-Level Versioning Details
- Rebuilding a clustered index is less likely to cause the non-clustered indexes to be rebuilt Details

Details

Large Rows

- Rows can now be larger than 8060 bytes with the row overflowing to other pages
  - Insert / update on a row with a variable length column that makes the row go over 8k in length will not fail

Details

tempdb Scalability

- Scalability of *tempdb* has been enhanced
  - Caching of initial pages for temporary tables and table variables
  - Improved allocation page latching protocol
  - Reduced logging overhead for *tempdb* so there is less I/O bandwidth for *tempdb* log file
  - More efficient allocation algorithm for mixed pages in *tempdb*

Details

tempdb Best Practices

- We still recommend the following if you see latch contention on *tempdb* allocation or system catalog pages:
  - Avoid auto grow
    - Pre-allocate space for *tempdb* files
  - Make as many *tempdb* files as you have CPUs
    - Account for any affinity mask settings
  - File sizes of equal amounts

Details

Backup

- Back up data and log concurrently
  - Scheduled Log Shipping backups are not blocked by a data backup
- RESTORE VERIFYONLY
  - Now does everything short of writing the restored pages to the disk
  - May take longer

Details

File I/O

- Page Checksum is on by default on new databases
  - Provides improved reliability
- Instant file initialization
  - Needs SE_MANAGE_VOLUME_NAME permission on service account
  - Instance-level configuration needed
  - Applies to CREATE DATABASE, ALTER DATABASE, RESTORE, file growth

Details

Insert Operations

- Insert Optimization
  - Write operations to contiguous blocks in the b-tree are optimized, as are heap inserts
  - Like bulk loading without the Bulk Load
  - Used by BCP into an existing table and on normal inserts working with a set of rows

Database Recovery

- Checkpoint
  - Fast without impacting foreground processes, continually adjusts itself
  - Parallel multi-database checkpoints
  - Interruptible
- Fast recovery
  - Database is available during undo phase
  - Enterprise Edition only

Row-Level Versioning

Details

- RCSI – Read Committed Snapshot Isolation
  - Requires database-level options to be enabled
    - ALLOW_SNAPSHOT_ ISOLATION and
    - READ_COMMITTED_SNAPSHOT
  - Isolation on transaction must be Read Committed
  - Override with Query hint: READCOMMITTEDLOCK
  - Improved concurrency and scalability; reduced deadlocks
- Triggers use Row-Level Versioning rather than accessing log records
  - Trigger execution more scalable
    - Avoids scans of log (which is often a bottleneck already)
    - Avoids duplicate scans for multiple connections
    - At least as fast as before; much faster, more scalable on high end
  - Used regardless of database's Snapshot Isolation settings

Clustered Index Rebuild

- Rebuilding / dropping a clustered index is less likely to cause the non-clustered indexes to be rebuilt
  - We preserve the uniqueifier so NCLIs remain correct
- For example, the following clustered index rebuilds would cause all NC indexes to be rebuilt, but not any more

```
CREATE TABLE tab ( a INT, b INT )
CREATE CLUSTERED INDEX idxc ON tab ( a )
CREATE NONCLUSTERED INDEX idxnc ON tab ( b )
INSERT INTO tab VALUES ( 1, 1 )
GO
DBCC DBREINDEX ( tab, idxc )
GO
CREATE CLUSTERED INDEX idxc ON tab ( a )
   WITH DROP_EXISTING
GO
ALTER INDEX idxc ON tab REBUILD -- new syntax
GO
```

SQL Server's Internal OS
SQLOS

- Dynamic Memory with AWE
- Buffer pool replacement policy
  - Even better at keeping the right pages in cache
- Constant Page Sniffer
  - Looks a random pages in buffer pool to see if the page's checksum has changed
    - Detects possible hardware problems
- DAC – Dedicated Admin Connection
- Deadlock output is in XML
- Load balance across CPUs differently
  - On larger machines threads were bound to session, now threads are bound to the batch instead
  - Many batches in a session might use other CPUs

Query Optimization

- Improved symbolic query simplifications
- "Non-updating updates" index maintenance optimization
- Indexed view matching improvements
  - We will match in more cases
- Predicate implication across equi-joins
  - R.x = S.x and S.x > 10 implies
    R.x = S.x and S.x > 10 and **R.x > 10**

Details